Pooling properties within the Graph Neural network framework

Luc Brun

Pooling properties within the Graph Neural network framework

luc Brun

May 2023

Abstract

Graph Neural Networks (GNNs) are inspired from CNNs and aim at transferring the performances observed on images to graphs. In a GNN, convolution and pooling are the main components in the network and these operations are employed in an alternating fashion between each other if a pooling method is used. However, this simple definition of GNN has some issues and their impacts can lead to low prediction performances. The two main issues are identified as *over-squashing* and *over-smoothing*. Recent works on these issues only focuses on the graph convolution operator, neglecting the role of pooling operator.

This paper aims to investigate the impact of pooling on over-squashing and over-smoothing. Our findings demonstrate that, under certain properties, pooling can reduced over-squashing and prevent over-smoothing. The conditions imposed on pooling to achieve these results are not so restrictive and encompass the majority of methods such as Top-k methods, EdgePool or MIS strategies. Finally, we empirically validate our results.

1 Introduction

Graph Neural Networks (GNNs) ([15]) take their inspiration from CNNs ([13]) and aim at transferring the performances observed on images to graphs. The backbone of CNNs and GNNs both rely on convolution and pooling operations. However, CNNs are designed for images which are embedded on a 2D or 3D oriented space and have a fixed structure. When working with graphs, one has to integrate several particularities. First, the number of nodes may vary between graphs. Second, the order of nodes for a given graph is generally arbitrary defined. A same graph can be represented by all the permutations of its node order, making complicate the comparison of two graphs. Third, unlike images, the structure of the graph is not regular and is defined by the neighborhood relationship between nodes. Therefore, the definition of GNNs must handle these problems to be effective on downstream tasks, and convolution and pooling operations must be adapted for graphs.

Similarly to convolutions in CNNs, first proposed graph convolution operators aim to learn node representations by defining a linear combination between

the representations of a given node and the representations of neighborhood nodes [11]. This convolution scheme is a particular case of a Message Passing Neural Network (MPNN) [8]. Following this, the new node representation is computed using an activation function, generally non linear to enhance the learning ability of the model. To compute a graph level representation, node representations of a graph are resumed into one single vector, using different strategies to keep all necessary information for the prediction tasks. This last step is generally denoted as the pooling operator, still inspired by CNNs.

However, this simple definition of GNN has some issues and their impacts can lead to low prediction performances. The two main issues are identified as *over-squashing* and *over-smoothing*.

Over-smoothing occurs when the number of layers, e.g. the depth, in a GNN increases. By iteratively combining neighbors node features together, all nodes representation in a graph are computed using the same information, hence leading to non discriminant node representations, and thus to a loss of information. We refer the reader to [14] for a survey on this issue.

Over-squashing is characterized by the fact that GNNs are almost unable to transfer information between distant nodes, hence limiting the learning possibilities of the model. This lack of communication between two nodes can be characterized by the length of their shortest paths, by commute (access) time ([9]), or by high negative curvature edges between them ([18]).

Graph pooling operators mainly consists in computing a new graph with a reduced number of nodes, while keeping as much as information as possible. The purpose of this operation is threefold : i) reduce the graph to one node so as the initial permutation of nodes has no influence in the learning process, ii) compute a hierarchy of graphs to catch relevant information and iii) enhance the receptive field of convolution operations. Unlike pooling in CNNs, this operation must be able to manage different graph sizes, nodes permutations and variations in local topologies. Graph pooling operators can generally be defined using the Select, Reduce and Connect scheme (SRC) [10]. These three operations respectively allows the definition of the set of nodes for the next level, how the new node features are computed and finally the structure of the new graph through the definition of edges on the new set of nodes.

In this paper, first, a study from a signal point of view on the need for subsampling and these limits is provided (Section 2). In Section 3, a decimation scheme is introduced to analyse the impact of restricting pooling as a partition of the initial graph on over-smoothing and over-squashing (Section 4). A complementary study is given in Section 5 for methods satisfying one of the conditions of the decimation scheme.

2 On the need for subsampling

The results bellow are mainly based on [1]. We consider a simple, connected, undirected and weighted graph G and denote by $\mathcal{L}$ its normalized Laplacian. The matrix $\mathcal{L}$ being symmetric positive semi-definite, we have $\mathcal{L} = U\Lambda U^T$

where U is the matrix of eigenvectors and $\Lambda = diag(\lambda_1, \ldots, \lambda_N)$ with $0 = \lambda_1, \lambda_2, \ldots, \lambda_N \leq 2$. Given a signal f on G, its Graph Fourier Transform (GFT) is defined as $\tilde{f} = U^T f$. A w-band limited signal on a graph is defined to have 0 GFT coefficients for frequencies above w, i.e. its spectral support is limited to $[0, w]$. The space of all w bandlimited signals is know as the *Paley-Wiener* space and is denoted $PW_w(G)$. Let us denote by $\{u_1, \ldots, u_N\}$ the columns of U. We trivially have

$$PW_w(G) = span(\{u_1, \ldots, u_i\}) \text{ with } \lambda_i \leq w.$$

Let us note that if $w < \lambda_2$ we have $PW_w(G) = \mathbb{R}u_1$ which corresponds to the set of constant signals on the graph.

Definition 1. *A subset of nodes $S \subset \mathcal{V}$ is a uniqueness set for the space $PW_w(G)$, if for any two signals from $PW_w(G)$, the fact that they coincide on S implies that they coincide on $\mathcal{V}$, i.e.:*

$$\forall f, g \in PW_w(G), \quad f(S) = g(S) \Rightarrow f = g$$

Let us note that for $w < \lambda_2$, any subset of $\mathcal{V}$ is a uniqueness subset of $PW_w(G)$. Indeed, for constant signals any sub-sampling even of a single vertex fully determines the signal on the whole graph.

Lemma 1. *S is a uniqueness set for a signal in $PW_w(G)$ iff $PW_w(G) \cap L_2(S^c) = \{0\}$, where $L_2(S^c)$ is the space of all graph signals that are zero everywhere except possibly on the subset of nodes S^c i.e.:*

$$\forall \phi \in L_2(S^c), \quad \phi(S) = 0$$

Proof. See [1]. □

Definition 2. *S is an allowed down-sampling for a signal f on G, if it exists w such that $f \in PW_w(G)$ and S is a uniqueness set for $PW_w(G)$.*

Let us consider the low pass filter g_n which cuts off (set to zeros) the n highest eigenvalues of $\mathcal{L}$. The filtered version of f, denoted f_n is equal to $f_n = U g_n(\Lambda) U^T f$. We have the following proposition:

Proposition 1. *For any signal f and for any uniqueness set S of a set $PW_w(G)$, it exists a filtered version f_n of f such that S is an allowed down-sampling for f_n.*

Proof. Let us consider $n = N - dim(PW_w(G))$. We have by definition of g_n, $f_n \in span(u_1, \ldots, u_{dim(PW_w(G))})$, hence $f_n \in PW_w(G)$. □

The above result shows that for any uniqueness set S and any signal f, a sufficient low pass filtering of f "forces" the set S to become an allowed down-sampling for the filtered version f_n of f. Intuitively, f_n contains a sufficient amount of redundant information to be described by the reduced set of vertices S. At the extreme, using $n = N - 1$, any subset $S \subset \mathcal{V}$ is an allowed down-sampling. It corresponds to the case where f has been filtered up to a constant signal. In this case, any sub-sampling of it is sufficient to describe the whole vector.

3 Sizes of receptive fields

Poolings operations define implicitly a hierarchy among vertices. Given a vertex v surviving at layer l we denote by $RW^l(v)$ the Reduction Window of v at level l, i.e. the set of vertices merged to v between levels $l-1$ and l. Let us note that we have $v \in RW^l(v)$. Say, an other way, $RW^l(v)$ denotes the child defined at level $l-1$ of v defined at level l. Given $u \in RW^l(v)$, we inversely say that v is the parent of u and denote this relation by $p^l(u) = v$, where $p^l : V^{l-1} \to V^l$ encodes the parent relationship.

We additionally suppose that the sequence of reduced graphs $G_0, \ldots, G_l$ is constrained by the reduction windows using the following equation:

$$\forall l, \forall (u,v) \in V^l, \quad d_{l-1}(RW^l(v), RW^l(u)) \leq 1 \Leftrightarrow u \in \mathcal{N}_l(v) \tag{1}$$

Where d_{l-1} is the distance defined within G_{l-1}. Stated in words, two adjacent reduction windows induce two adjacent vertices.

The transitive closure of the hierarchy relationships defined by the reduction window defines the receptive field. The receptive field of v at level l $RF^l(v)$ corresponds to the set of vertices defined at the base level graph which are merged onto v at level l. More formally, the receptive field at level l is defined recursively :

Definition 3. *Let $G_l, \ldots, G_1 = (V^1, E^1)$ denote a sequence of reduced graphs. The receptive fields at level l are defined for any vertex $v \in V^l$ as:*

$$RF^l(v) = \bigcup_{u \in RW^l(v)} RF^{l-1}(u) \text{ with } RF^1(u) = RW^1(u)$$

Equation 1 can be extended to receptive fields:

Proposition 2. *Let $G_l, \ldots, G_1 = (V^1, E^1)$ denote a sequence of reduced graphs:*

$$\forall l, \forall (u,v) \in V^l, \quad d_0(RF^l(v), RF^l(u)) \leq 1 \Leftrightarrow u \in \mathcal{N}_l(v)$$

Proof. The property is true for $l = 1$ since at this level receptive fields are equal to reduction windows. Let us suppose it true up to level $l-1$, and let us consider two vertices u and v such that $d(RF^l(v), RF^l(u)) \leq 1$. We obtain:

$$d_0\left(\bigcup_{v' \in RW^l(v)} RF^{l-1}(v'), \bigcup_{u' \in RW^l(u)} RF^{l-1}(u')\right) \leq 1$$

Hence, it must exists $u' \in RW^l(u)$ and $v' \in RW^l(v)$ such that $d_0(RF^{l-1}(v'), RF^{l-1}(u')) \leq 1$. We thus get by our recurrence hypothesis: $u' \in \mathcal{N}_{l-1}(v')$. We thus obtain:

$$d_{l-1}(RW^l(v), RW^l(u)) \leq d_{l-1}(u', v') = 1$$

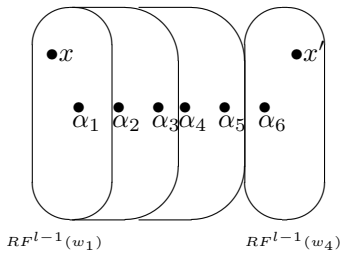

Figure 1: Illustration of Corollary 1 with $q = 4$.

and thus $u \in \mathcal{N}_l(v)$.

Conversely, if $u \in \mathcal{N}_l(v)$, we have $d_{l-1}(RW^l(v), RW^l(u)) \leq 1$. So it must exists $(u', v') \in RW^l(u) \times RW^l(v)$ such that $d_{l-1}(u', v') \leq 1$.

If $u' = v'$, then $d_0(RF^l(u), RF^l(v)) = 0$. Otherwise, $u' \in \mathcal{N}_{l-1}(v')$. By iterating this process we can get two sequences $u^0 = u, \ldots, u^l \in V^0$ and $v^0 = v, \ldots, v^l \in V^0$ such that $u^l \in \mathcal{N}_0(v^l)$ and u is an ancestor of u^l while v is an ancestor of v^l. We have thus $u^l \in RF^l(u)$ and $v^l \in RF^l(v)$ with $d_0(u^l, v^l) \leq 1$. Hence the expected result. □

The above proposition provides the following interesting corollary:

Corollary 1. *Let $G_l, \ldots, G_1 = (V^1, E^1)$ denote a sequence of reduced graphs, if all reduction windows (at any level) are connected, then receptive fields are also connected.*

Proof. The proof is trivial at level 1, since at this level, receptive fields are equal to reduction windows. Let us suppose it true up to level $l-1$. We have for any vertex $v \in V^l$:

$$RF^l(v) = \bigcup_{u \in RW^l(v)} RF^{l-1}(u)$$

Given any pair of vertex x, x' in $RF^l(v)$, let us consider w and w' in $RW^l(v)$ such that $x \in RF^{l-1}(w)$ and $x' \in RF^{l-1}(w')$. If $w = w'$, we have nothing left to demonstrate since $RF^{l-1}(w)$ is connected by hypothesis. Otherwise, since $RW^l(v)$ is connected, it exists a path $w_1 = w, \ldots, w_q = w'$ connecting w and w' in $RW^l(v)$. We have by definition of a path: $w_{i-1} \in \mathcal{N}_{l-1}(w_i)$ for any i in $\{2, \ldots, q\}$. Hence, using proposition 2, $d_0(RF^{l-1}(w_{i-1}), RF^{l-1}(w_i)) \leq 1$, for any $i \in \{2, \ldots, q\}$. We can thus identify a set of vertices $\alpha_1, \ldots, \alpha_{2q-2}$ in V^0 such that $\alpha_1 \in RF^{l-1}(w_1), \alpha_{2q-2} \in RF^{l-1}(w_q)$, $\{\alpha_{2i-2}, \alpha_{2i-1}\} \subset RF^{l-1}(w_i)$ for $i \in \{2, \ldots, q-1\}$ and $d_0(\alpha_{i-1}, \alpha_i) \leq 1$, for any $i \in \{2, \ldots, q\}$ (Figure 1). Since all $RF^{l-1}(w_i)$ are connected, it exists a path $x, \alpha_1, \ldots, \alpha_{2q-2}x'$ connecting x and x' in $RF^l(v)$. This last set is thus connected. □

If the set of reduction windows defined at any level l defines a partition of G_{l-1}, the set of receptive fields defined at any level l defines a partition of G_0.

Moreover, if $RW^l(w) = \{v_1, \ldots, v_n\}$, then $\{RF^{l-1}(v_1), \ldots, RF^{l-1}(v_n)\}$ forms a partition of $RF^l(w)$.

We proposed in [16, 17] different strategies to define a graph hierarchy. Reduction windows produced by these strategies satisfy the following equations at any layer l and for any vertex $w \in V^l$:

$$\begin{cases} RW^l(w) & = & \{w\} \text{ or} \\ RW^l(w) & = & \{w, v_1, \ldots, v_n\} \text{ with } \forall i \in \{1, \ldots, n\}\ d_{G_{l-1}}(w, v_i) = 1 \end{cases} \tag{2}$$

where $d_{G_{l-1}}(.,.)$ is the distance within the graph G_{l-1} defined at layer $l-1$.

Let us first note that this restrictions on reduction windows induces connected receptive fields (Corollary 1). We may additionally note that equation 2 is satisfied by the methods [16, 17] but also by all methods based on a maximal matching [5, 6, 12]. However dense methods based on clustering [21, 2, 22] algorithms generally do not satisfy these conditions. Finally, Top-k methods satisfy only the first condition ($RW^l(w) = \{w\}$). It corresponds to a special case which is considered in Section 5.

Proposition 3. *Using a decimation scheme satisfying equation 2 we have for any vertex w surviving at level l in the hierarchy:*

$$\forall (u, v) \in RF^l(w)^2 \quad d_{G_0}(u, v) \leq 2 * 3^l - 1$$

Proof. For $l = 1$, since any non surviving vertex is connected to a surviving one, all paths do not exceed 2.

Let us suppose the property true up to layer $l-1$ and let us consider a vertex w surviving up to level l. If $RW^l(w) = \{w\}$ the proof is trivial. Otherwise, $RW^l(w) = \{w, v_1, \ldots, v_n\}$. Using equation 2, $\{v1, \ldots, v_n\}$ are adjacent to w. Let us consider two vertices u and v and two indexes r and s such that $u \in RF^{l-1}(v_r)$ and $v \in RF^{l-1}(v_s)$. Since v_r and v_s are both connected to w, their receptive fields are adjacents to the one of w. It exists thus a path $P_{uv} = P_{v_r} P_w P_{v_s}$ connecting u and v such that $P_t \subset RF^{l-1}(t), t = v_r, v_s, w$ (Figure 2). We have thus:

$$d_{G_0}(u, v) \leq |P_{uv}| \leq \sum_{t \in \{v_r, w, v_z\}} |P_t| + 2 \leq 3(2 * 3^{l-1} - 1) + 2 = 2 * 3^l - 1$$

The two other cases (v_r or v_s equal to w or both u and v belong to $RF^{l-1}(w)$) imply respectively 2 or 1 subpaths. They thus also satisfy the inequality. □

4 Pooling operations

In the following, we suppose that we do not discard information between levels. More precisely, at any level a vertex either survives or is attached to an unique surviving neighbor.

Let us consider the simple following pooling function:

$$h^l = pool^l(h_{l-1}) = S^l h_{l-1} \tag{3}$$

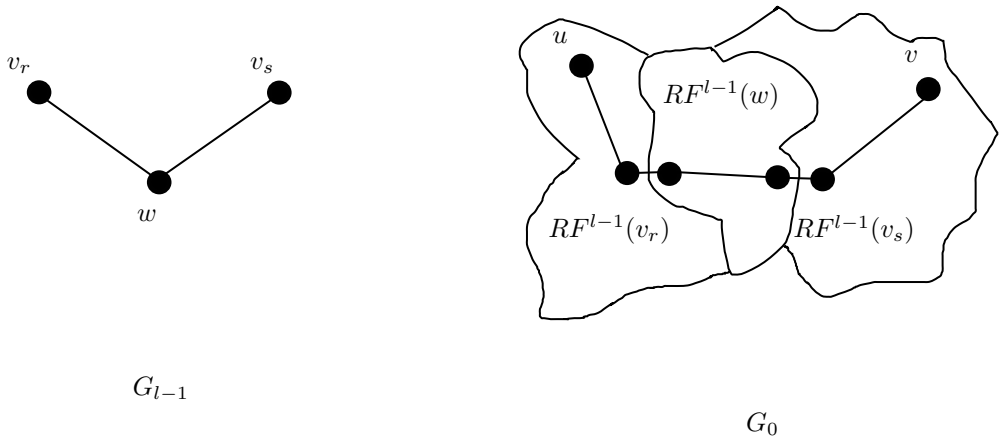

Figure 2: Connection between receptive fields in the base level graph. Note the two edges between $RF^{l-1}(v_r)$ and $RF^{l-1}(w)$ and between $RF^{l-1}(w)$ and $RF^{l-1}(v_s)$.

Iterating the previous equation up to level 0 lead to:

$$h^q = \left(\prod_{l=1}^{q} S^l\right) h^0 =_{not} \Sigma^q h^0$$

where Σ^q denotes the product $\prod_{l=1}^{q} S^l \in \mathbb{R}^{n_q \times n}$, where n_q is the number of vertices of G_q and n the number of vertices of G_0.

Proposition 4. *Using previous notations, if we denote by $\sigma^q_{i,j}$ the entry (i,j) of Σ^q, we have:*

$$\sigma^q_{i,j} = s^q_{i,p^{q-1}(j)} \left(\prod_{k=1}^{q-1} s^k_{p^k(j),p^{k-1}(j)}\right)$$

where $s^k_{m,n}$ is the (m,n) entry of S^k, and $p^k(j)$ is the parent of j at level k. By convention $p^0(j) = j$.

Let us note that $s^q_{i,p^{q-1}(j)} \neq 0$ iff $p^{q-1}(j)$ is merged with i at level q and we have in this case $p^q(j) = i$. So the previous equation may be rewritten:

$$\sigma^q_{i,j} = \begin{cases} 0 & \text{If } p^q(j) \neq i \\ \prod_{k=1}^{q} s^k_{p^k(j),p^{k-1}(j)} & \text{oherwize} \end{cases}$$

Proof. Let us first note the asymmetry between i and j in Proposition 4. By hypothesis, i survives at level q while j is just one vertex of G_0 which may have been contracted at any level between level 0 and level q.

Let us consider the case $q = 2$. We have:

$$\sigma^2_{i,j} = (S^2 S^1)_{i,j} = \sum_{k=1}^{n} s^2_{i,k} s^1_{k,j}$$

By hypothesis, the vertex j either survives or is merged with a single survivor. In any case, there is a single k (the parent $(p^1(j))$ of j) such that $s^1_{k,j} \neq 0$. We thus have:

$$\sigma^2_{i,j} = s^2_{i,p^1(j)} s^1_{p^1(j),j} = s^2_{i,p^1(j)} s^1_{p^1(j),p^0(j)}$$

Let us note that $\sigma^2_{i,j} \neq 0$, only if $s^2_{i,p^1(j)} \neq 0$ and hence if $p^1(j) \in RW^2(i)$. Let us suppose the property true up to level $q-1$, we have:

$$\sigma^q_{i,j} = \sum_{k=1}^{n_{q-1}} s^q_{i,k} \sigma^{q-1}_{k,j}$$

By recurrence hypothesis, $\sigma^{q-1}_{k,j} \neq 0$, only if $s^{q-1}_{k,p^{q-2}(j)} \neq 0$ which implies that $p^{q-2}(j) \in RW^{q-1}(k)$ and hence that $k = p^{q-1}(j)$. We have thus:

$$\sigma^q_{i,j} = s^q_{i,p^{q-1}(j)} \sigma^{q-1}_{p^{q-1}(j),j} = s^q_{i,p^{q-1}(j)} s^{q-1}_{p^{q-1}(j),p^{q-2}(j)} \left(\prod_{k=1}^{q-2} s^k_{p^k(j),p^{k-1}(j)} \right)$$

which provides the expected equation, namely:

$$\sigma^q_{i,j} = s^q_{i,p^{q-1}(j)} \left(\prod_{k=1}^{q-1} s^k_{p^k(j),p^{k-1}(j)} \right)$$

Moreover, $\sigma^q_{i,j} \neq 0$ only if $s^q_{i,p^{q-1}(j)} \neq 0$. □

Let us note that we have $h^q = \Sigma^q h^0$. So, let us introduce $u^q = \Sigma^q \mathbb{1}_n$, where $\mathbb{1}_n \in \mathbb{R}^n$ denotes the vector of ones. u^q, encodes for each vertex surviving at level q, the sum of the weighs of the vertices defined at level 0 which contribute to its value.

Proposition 5. *If S^q is line stochastic for any layer q then:*

$$\forall q \; u^q = \mathbb{1}_{n_q}$$

Proof. Matrix Σ^q is line stochastic as a product of line stochastic matrices. Moreover, we have, $u^0 = \mathbb{1}_n$ since each vertex represents itself in the base level graph. The result follows. □

Note that, the same demonstration than Proposition 5, can state that if the rows of S^q sum to γ for any q, then we have $u^q = \gamma^q \mathbb{1}_{n_q}$. We observe then either a drastic decrease ($\gamma < 1$)) or an explosion ($\gamma > 1$) of the features of the reduced graphs.

4.1 Over smoothing

As mentioned in Proposition 4, $\sigma^q_{i,j} \neq 0$, only if $p^q(j) = i$ i.e. $j \in RF^q(i)$. Hence:

$$h^q = \Sigma^q h^0 \Rightarrow h^q_i = \sum_{j=1}^{n_q} \sigma^q_{i,j} h^0_j = \sum_{j \in RF^q(i)} \sigma^q_{i,j} h^0_j \tag{4}$$

Each entry h_i^q is thus a weighted mean (Proposition 5) of the vertices defined at the base level and belonging to the receptive field of i. Since the set of receptive fields forms a partition of the initial vertex set, having two vertices at level q with a same value would imply that two different linear combinations of two sets of different values result in a same value. This possibility is quite reduced.

4.2 Over squashing

Using previous notations, we denote by $h_i^{(q,\alpha)}$ the component α of the feature vector of i defined at level q. We also consider a vertex j whose sequence of parents up to level q is defined by $p^0(j) = j, p^1(j), \ldots, p^q(j)$. We finally consider the layer $m \leq q$ where a parent of j merges with i (so $p^m(j) = i$). Since i survives up to layer q, we have $p^0(i) = p^1(i) = \ldots = p^q(i) = i$ (Figure 4).

If we suppose Σ^q independent of h^0, we obtain from equation 4:

$$\frac{\partial h_i^{(q,\alpha)}}{\partial h_j^{(0,\beta)}} = \sigma_{i,j}^q \delta_{\alpha,\beta} = s^q_{i,p^{q-1}(j)} \left(\prod_{k=1}^{q-1} s^k_{p^k(j),p^{k-1}(j)} \right) \delta_{\alpha,\beta}$$

where δ is the dirac function.

Since $p^m(j) = \ldots = p^q(j) = i$ we obtain:

$$\frac{\partial h_i^{(q,\alpha)}}{\partial h_j^{(0,\beta)}} = \left(\prod_{k=m+1}^{q-1} s^q_{i,i} \right) \left(\prod_{k=1}^{m} s^k_{p^k(j),p^{k-1}(j)} \right) \delta_{\alpha,\beta}$$

Note that for $k < m$, $j \notin RF^k(i)$, hence $\sigma_{i,j}^k = 0$ and $\frac{\partial h_i^{(k,\alpha)}}{\partial h_j^{(0,\beta)}} = 0$.

As an exercise, let us suppose that $s^k_{ww} = 1$ for any vertex w and any layer k. Let us also denote by r the layer where j dies. We get:

$$\frac{\partial h_i^{(q,\alpha)}}{\partial h_j^{(0,\beta)}} = \left(\prod_{k=r+1}^{m} s^k_{p^k(j),p^{k-1}(j)} \right) \delta_{\alpha,\beta}$$

In this case, only merge operations induce a decrease of the gradient and the derivative $\frac{\partial h_i^{(k,\alpha)}}{\partial h_j^{(0,\beta)}}$ remains constant from layer m up to layer q.

The different behaviors of $\frac{\partial h_i^{(q,\alpha)}}{\partial h_j^{(0,\beta)}}$ according to both scenarios ($s^k_{ww} = 1$ and $s^k_{ww} < 1$) is provided in Figure 5.

Let us now bound the level m where i and j merge. Using the same notation, m is the first level such that $j \in RF^m(i)$. So, if the decimation satisfies the hypothesis defined by equation 2, we have (Proposition 3):

$$d_{G_0}(i,j) \leq 2 * 3^m - 1 \Rightarrow m \geq \log_3 \left(\frac{d_{G_0}(i,j) + 1}{2} \right) \tag{5}$$

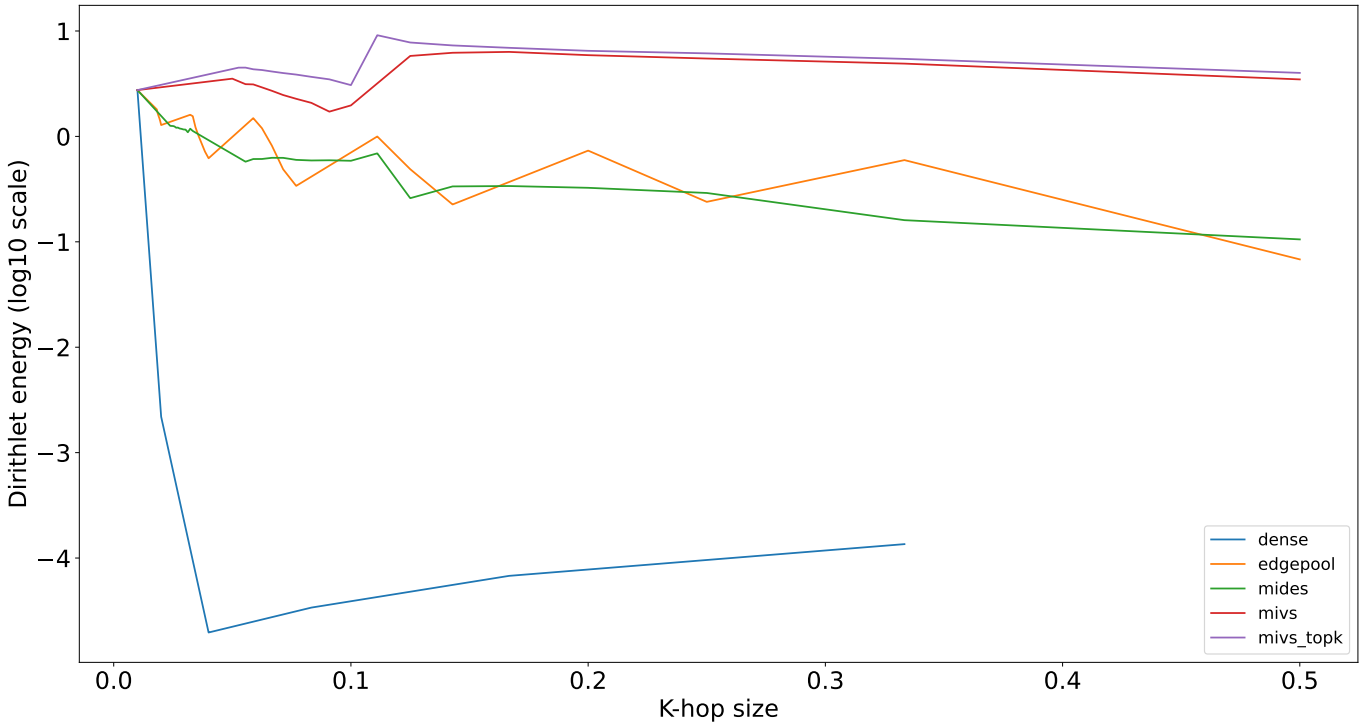

Figure 3: Let consider 1000 graphs with 100 nodes and a density of 20% generating like in [7]. The curves shows the mean Dirichlet energy according to the size of K-hop. *dense*, *edgepool*, *mides*, *mivs* and *mivs_topk* respectively denote an architecture only composed of DiffPool [21], EdgePool [6], MIDESPool [17], MIVSPool [16] and Top-k where the selection of nodes are conditionned by a Maximal Independent Vertex Set and the connection of surviving nodes is the same as MIVSPool. Note that any learnable parameters is replaced by random values.

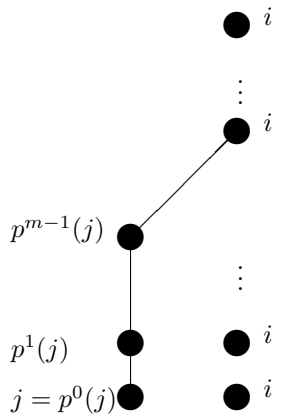

Figure 4: Sequences of parents of nodes u and v.

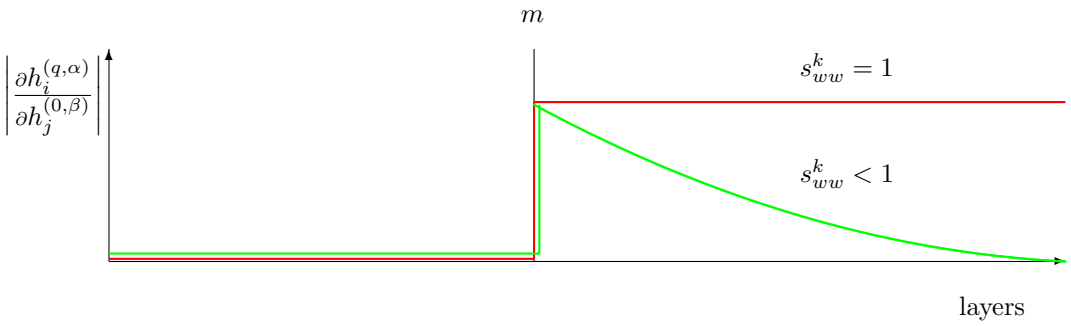

Figure 5: Different behaviors of the derivative of $h_v^{(l,\alpha)}$ according to $h_u^{(0,\alpha)}$ according to to the weight of the surviving vertex.

The minimal number of layers required to "connect" any two vertices is thus a log of the graph diameter. For example, at least 2 layers are required to group two vertices whose distance is equal to 17.

In order to obtain a lower bound for m, we should make additional hypothesis. Before this, let us consider two vertices x and y surviving up to level r and let us consider two paths such that:

$$\begin{cases} |P_{xy}^{r}| &= d_{G_r}(x,y) \quad \text{and} \\ |P_{xy}^{r-1}| &= d_{G_{r-1}}(x,y) \end{cases}$$

Let us now consider the path Q_{xy}^{r-1} formed from P_{xy}^{r-1} by contracting all the edges concerned with the transformation of G_{p-1} onto G_p present in P_{xy}^{r-1}. Q_{xy}^{r-1} is a sequence of adjacent surviving vertices at level r and is thus a valid path of G_r. Moreover, Q_{xy}^{r-1} is trivially shorter than P_{xy}^{r-1}. We have thus:

$$d_{G_{r-1}}(x,y) = |P_{xy}^{r-1}| \geq |Q_{xy}^{r-1}| \geq |P_{xy}^{r}| = d_{G_r}(x,y)$$

The above equation, only stress a trivial result, namely the fact that contractions shrink distances. Let us go one step further by supposing the existence of a constant $\gamma > 1$ such that for any level p and any couple of surviving vertices (x,y) we have:

$$d_{G_{p-1}}(x,y) \geq \gamma d_{G_p}(x,y)$$

We hence suppose a small positive multiplicative margin. In this case we have:

$$d_{G_p}(x,y) \leq \frac{1}{\gamma} d_{G_{p-1}}(x,y) \leq \left(\frac{1}{\gamma}\right)^{p} d_{G_0}(x,y)$$

Returning to i and j, let us additionally suppose that j survives up to level $m-1$ where it is merged with i. We have thus $p^0(j) = \ldots = p^{m-1}(j) = j$ and $p^m(j) = \ldots = p^q(j) = i$. Since i and j merge at level m, it means that they are neighbors at level $m-1$ (equation 2). We thus have:

$$d_{G_{m-1}}(i,j) = 1 \leq \left(\frac{1}{\gamma}\right)^{m-1} d_{G_0}(i,j) \Rightarrow \gamma^{m-1} \leq d_{G_0}(i,j)$$

We thus obtain:

$$m \leq \frac{\log(d_{G_0}(i,j))}{\log(\gamma)} + 1 \tag{6}$$

One may note the similarities between equations 5 and 6 which provide respectively a lower and an upper bound for m both according to $\log(d_{G_0}(i,j))$.

We can thus assume that any two vertices are connected within our network at a layer roughly proportional to the *log* of the distance between these vertices. This is important since it means that contrary to usual graph neural networks without poolings, which require a linear number of layers according to the distance between the two vertices to be connected, a GNN based on a decimation scheme fulfilling equation 2 connects them much faster. All the problematic, connected to the attenuation of the influence of u on v along a linear number of layers is thus strongly reduced.

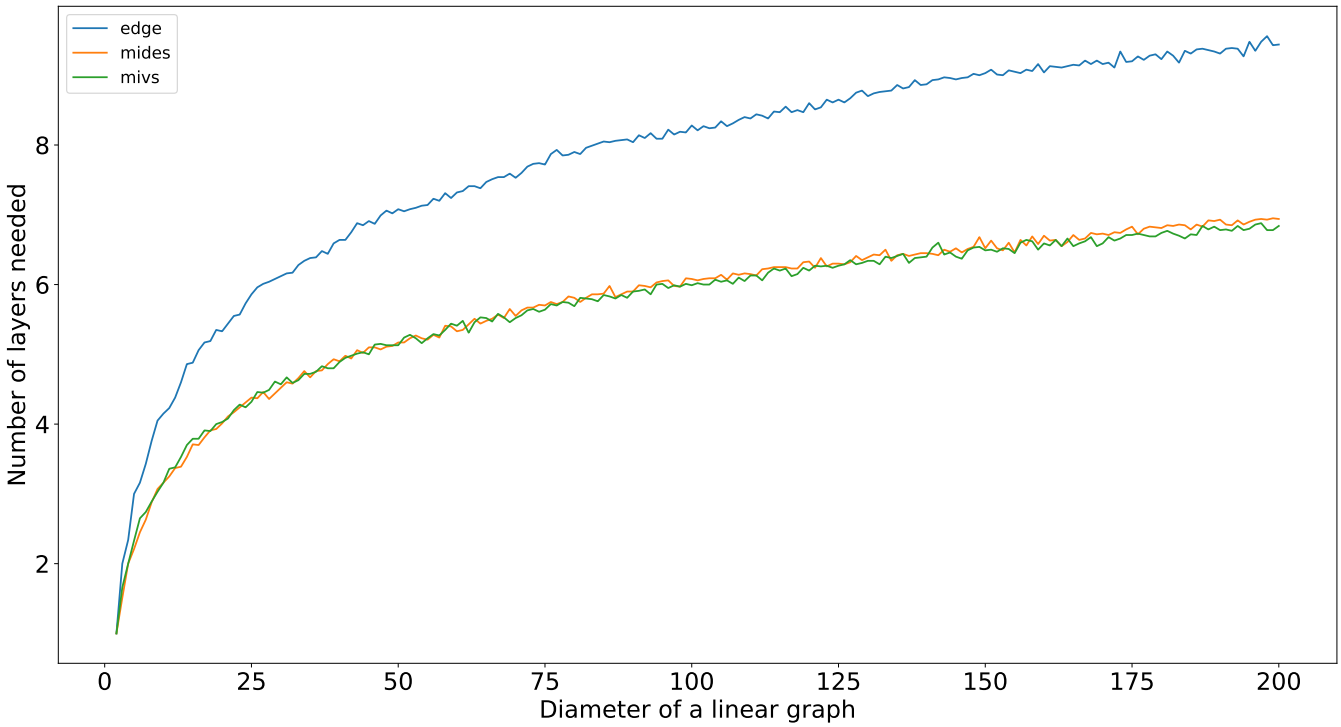

Figure 6: Let consider a linear graph with features (1,0) and (0,1) at the extreme nodes. Others nodes features are equal to (0,0). In the figure, we study the average number of layers needed to intersect the two features of the extreme nodes according to the diameter of the graph. *edge*, *mides* and *mivs* respectively denote an architecture only composed of EdgePool [6], MIDESPool [17] and MIVSPool [16]. Note that any learnable parameters is replaced by random values.

5 On Top-k methods and Graph Shift Operators

Top-k methods perform a selection of vertices among the graph before reconnecting it. Such a selection procedure may be modeled at layer l by a matrix $S \in \mathbb{R}^{n_{l+1} \times n_l}$ where each row has a single positive entry. Columns may contain either a single entry (of a surviving vertex) or be filled with 0 (the non surviving vertex is discarded).

Definition 4. *A matrix O is called a Graph Shift Operator (GSO) for a graph $G_=(V, E)$ iff:*

$$\forall (i,j) \in V^2, \quad i \neq j \text{ and } (i,j) \notin E \Rightarrow o_{i,j} = 0$$

Graph Shift operators [4] are taken are a generalization of adjacency matrices and are used to generalize different kind of convolutions.

Proposition 6. *Given a graph G, a GSO $O \in \mathbb{R}^{n \times n}$, and a projection matrix $S \in \mathbb{R}^{n' \times n}$ with $n' < n$, then the product matrix SO satisfies:*

$$\forall (i,j) \in V' \times V, \quad k_i \neq j \text{ and } (k_i, j) \notin E \Rightarrow so_{i,j} = 0$$

Where V' is the reduced set of vertices with $|V'| = n'$ and $k_i \in V$, is the unique son of $i \in V'$.

Proof. Since S has a single non null value per line, multiplying O per S is equivalent to select the lines of O corresponding to the non null column entries of S and to multiply them by the corresponding entry. In other terms we have:

$$so_{i,j} = s_{i,k_i} o_{k_i,j} = s_i o_{k_i,j}$$

where s_i denotes the value of the only positive entry on line i and k_i is the index selected at line i. The proof follows from the fact that O is a GSO.

□

Proposition 6, indicates that SO, is not a GSO since it performs a shift and a selection. Consequently the matrix SO is not squared. However, the main property of a GSO is preserved.

Let us consider a generic form of GSO, denoted as the *Parametrised Graph Shift Operator* (PGSO), defined by the following expression [4]:

$$\gamma(A, \mathcal{S}) = m_1 D_a^{e_1} + m_2 D_a^{e_2} A_a D_a^{e_3} + m_3 I$$

where $A_a = A + aI$, $D_a = diag(A_a 1)$ is the degree matrix of A_a and $\mathcal{S} = (m_1, m_2, m_3, e_1, e_2, e_3, a)$ is our set of parameters.

We have:

$$S\gamma(A, \mathcal{S}) = m_1 S D_a^{e_1} + m_2 S D_a^{e_2} A_a D_a^{e_3} + m_3 S$$

Let us note that that $SD_a^{e_1}$ just multiplies the non null entry of each line i of S by $(d_a^{e_1})_{k_i}$. The main properties of our projection (a single non null entry on each

line) is thus preserved and $SD_a^{e_1}$ may be considered as a modified projection. So that we can write:

$$S\gamma(A,\mathcal{S}) = S_1 + S_2 A D_a^{e_3}$$

where $S_1 = S(m_1 D_a^{e_1} + a m_2 D_a^{e_2+e_3} + m_3)$ and $S_2 = m_2 S D_a^{e_2}$ are two projections.

Considering the equation $h' = S\gamma(A,\mathcal{S})h$ we obtain by dropping the sub index 'a':

$$h'_i = s_i \left[(m_1 d_{k_i}^{e_1} + a m_2 d_{k_i}^{e_2+e_3} + m_3) h_{k_i} + m_2 d_{k_i}^{e_2} \sum_{j \in \mathcal{N}_{k_i}} a_{k_i,j} d_j^{e_3} h_j \right]$$

where (d_i) stands for the entries of D_a and $(a_{i,j})$ for the ones of A. Let us recall that s_i denotes s_{i,k_i} the only positive entry of line i.

Using projections matrices S_1 and S_2 we get the simpler formula:

$$h'_i = s_i^1 h_{k_i} + s_i^2 \sum_{j \in \mathcal{N}_{k_i}} a_{k_i,j} d_j^{e_3} h_j \tag{7}$$

The main difference between equations 3 and 7 is that the pooling step defined in equation 3 requires that each non surviving vertex is attached to a unique survivor while a non surviving vertex in equation 7 shares its value with all surviving vertices adjacent to it. In terms of matrices, it means that the matrix S^l, used in equation 3 has a single positive entry per column while if we denote by S^{top} the matrix implicitly defined by equation 7, so that $h' = S^{top}h$, each column of S^{top} corresponding to a non surviving vertex has one positive entry per surviving vertex adjacent to the non surviving one.

5.1 Some examples

Let us consider the simplest PGSO defined by $\gamma(A,\mathcal{S}) = I + A$. It corresponds to $\mathcal{S} = (m_1, m_2, m_3, e_1, e_2, e_3, a) = (0, 1, 1, 0, 0, 0, 0)$. We then have $S_1 = S$ and $S_2 = S$. We thus obtain:

$$h'_i = s_i \left[h_{k_i} + \sum_{j \in \mathcal{N}_{k_i}} a_{k_i j} h_j \right] \tag{8}$$

where k_i denotes the only positive entry of S^{l+1} in row i and $\mathcal{N}_{k_i}$ denotes the neighborhood of k_i at layer l.

Equation 8, has two major normalisation problems:

1. A non surviving vertex j transfers its values to all its surviving neighbors. Hence, non surviving vertices with a high degree will be over-represented compared to surviving ones or non surviving ones with a smaller degree.

2. The sum in equation 8 is proportional to $|\mathcal{N}_{k_i}|$. Hence surviving vertices with a large degree will be over weighted according to the other surviving vertices. Large neighborhood may also hide the value $h_{k_i}^l$ of the surviving vertex at layer l.

In order to overcome the two previous drawbacks, we may consider the following PGSO defined by $\mathcal{S} = (0, 1, 1, 0, -1, -1, 0)$ which provides the following update equation:

$$h' = S(I + D^{-1}AD^{-1})h$$

In this case, we obtain:

$$h'_i = s_i \left[h_{k_i} + \frac{1}{d_{k_i}} \sum_{j \in \mathcal{N}_{k_i}} \frac{a_{k_i j}}{d_j} h_j \right]$$

The main drawback of the last equation is that the contributions of non surviving vertices is normalized by their degree and not by their number of surviving neighbors (to which they contribute). To correct this point we first note that $(SA)_{ij} = s_{ik_i} a_{k_i j}$ so that:

$$((SA)^T 1)_j = \sum_{i=1}^{n_{l+1}} s_{ik_i} a_{k_i j} = \sum_{i \in V^{l+1} | k_i \in \mathcal{N}_j} s_{ik_i}$$

So, $((SA)^T 1)_j$ is the sum of the weighs (or the scores) of the surviving vertices adjacent to each vertex j. Let us consider the diagonal matrix $D^{surv} = diag((SA)^T 1)$. In order to incorporate D^{surv} we need to mix pooling and convolution steps by considering the following update equation:

$$h_{l+1} = S \left(I + D^{-1} A (D^{surv})^{-1} \right) h^l$$

We then get:

$$h_i^{l+1} = s_{ik_i} \left[h_{k_i}^l + \frac{1}{d_{k_i}} \sum_{j \in \mathcal{N}_{k_i}} \frac{a_{k_i j}}{d_j^{surv}} h_j^l \right]$$

Note that $\frac{s_{ik_i}}{d_j^{surv}}$ may be interpreted as the relative importance of the surviving vertex k_i among the neighbors of j. We have thus a strategy of "winner takes all".

5.2 Effect on main convolutions

Let us consider the PGSO of GCN [11] defined by $\gamma(A, \mathcal{S}) = D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}}$. It corresponds to $\mathcal{S} = (m_1, m_2, m_3, e_1, e_2, e_3, a) = (0, 1, 0, 0, -\frac{1}{2}, -\frac{1}{2}, 1)$. We then have $S_1 = SD_1^{-1}$ and $S_2 = SD_1^{-\frac{1}{2}}$. We thus obtain:

$$h'_i = s_i d_{k_i}^{-1} h_{k_i} + s_i d_{k_i}^{-\frac{1}{2}} \sum_{j \in \mathcal{N}_{k_i}} a_{k_i, j} d_j^{-\frac{1}{2}} h_j \tag{9}$$

where k_i denotes the only positive entry of S^{l+1} in row i and $\mathcal{N}_{k_i}$ denotes the neighborhood of k_i at layer l.

Let us consider the PGSO of GIN [20] defined by $\gamma(A, \mathcal{S}) = A + (1+\epsilon)I$. It corresponds to $\mathcal{S} = (m_1, m_2, m_3, e_1, e_2, e_3, a) = (0, 1, 0, 0, 0, 0, 1+\epsilon)$. We then have $S_1 = (1+\epsilon)S$ and $S_2 = S$. We thus obtain:

$$h'_i = (1+\epsilon)s_i h_{k_i} + s_i \sum_{j \in \mathcal{N}_{k_i}} a_{k_i,j} h_j \tag{10}$$

where k_i denotes the only positive entry of S^{l+1} in row i and $\mathcal{N}_{k_i}$ denotes the neighborhood of k_i at layer l.

Let us consider the PGSO of DGCNN [23] defined by $\gamma(A, \mathcal{S}) = D_1^{-1} A_1$. It corresponds to $\mathcal{S} = (m_1, m_2, m_3, e_1, e_2, e_3, a) = (0, 1, 0, 0, -1, 0, 1)$. We then have S_1 and S_2 both equal to SD_1^{-1}. We thus obtain:

$$h'_i = s_i d_{k_i}^{-1} h_{k_i} + s_i d_{k_i}^{-1} \sum_{j \in \mathcal{N}_{k_i}} a_{k_i,j} h_j \tag{11}$$

where k_i denotes the only positive entry of S^{l+1} in row i and $\mathcal{N}_{k_i}$ denotes the neighborhood of k_i at layer l.

Convolution	$\mathcal{S}$	$\gamma(A, \mathcal{S})$
GCN	$(0, 1, 0, 0, -\frac{1}{2}, -\frac{1}{2}, 1)$	$D_1^{-\frac{1}{2}} A_1 D_1^{-\frac{1}{2}}$
GIN	$(0, 1, 0, 0, 0, 0, 1+\epsilon)$	$A + (1+\epsilon)I$
DGCNN	$(0, 1, 0, 0, -1, 0, 1)$	$D_1^{-1} A_1$

Table 1: Summary of the convolution formulas

Let note that PGSO can't define all graph convolutions. For example, GAT methods [19, 3] can't be define without additional assumptions or operators.

5.3 Let us iterate

Let's assume that the matrix A has a non-zero diagonal and let us consider the matrix $S^{top} = S\gamma(A, S)$. Let note that, using Proposition 6 S^{top} satisfies the main properties of a GSO while being a non squared matrix. Let us now consider the two following iteration equations:

$$\left\{ \begin{array}{lcl} h' & = & S^{top} h \\ A' & = & S^{top} A (S^{top})^T \end{array} \right. \tag{12}$$

Where A is the adjacency matrix with positive elements on the diagonal. Let us simply denote by $s_{i,j}$ the entries of the matrix S^{top} (hence dropping the top index). Then we have:

$$a'_{ij} = \sum_{k,l=1}^{n} a_{k,l} s_{ik} s_{jl}$$

If A and S^{top} are non negative, we get the trivial result:

$$a'_{i,j} \neq 0 \text{ iff } \exists (k,l) \in \{1, \ldots, n\}^2 \quad | \quad \left\{ \begin{array}{lcll} (k,l) & \in & E & (a_{kl} \neq 0) \\ s_{ik} & \neq & 0 & \\ s_{jl} & \neq & 0 & \end{array} \right. \tag{13}$$

This general equation, hides some particular cases. In particular, a vertex shared by two surviving vertices induces an edge between them:

$$\exists k \in \{1, \dots, N\} \,|\, s_{ik} \neq 0 \text{ and } s_{j,k} \neq 0 \Rightarrow a'_{i,j} \neq 0$$

The above equation is simply due to the fact that a_{kk} is always not null. Moreover, we have for any reduced vertex i:

$$a'_{i,i} \geq a_{k_i,k_i} s_{i,k_i} s_{i,k_i} > 0$$

Let us now introduce reduction windows within the Top-k framework.

Definition 5. *Let $G = (A, h)$ denote a graph and $G' = (A', h')$ its reduced version using equations 12 with $S^{top} \in \mathbb{R}^{n' \times n}$. The reduction window of a surviving vertex $i \in \{1, \dots, n'\}$ is defined by:*

$$RW(i) = \{k \in \{1, \dots, n\} \quad | \quad s_{ik} > 0\}$$

Note that reduction windows defined with the classical pooling framework may be defined using the same equation than in Definition 5. The main difference within the Top-k framework is that we loss the insurance that each column of S^{top} contains a single positive entry. Consequently, a non surviving vertex may belong to different reduction windows and the set of reduction window may not form a partition of $\{1, \dots, n\}$.

Remark 1. *Using Proposition 6 we have for any surviving vertex i:*

$$RW(i) \subset \mathcal{N}_{k_i} \cup \{k_i\}$$

where $\mathcal{N}_{k_i}$ denotes the neighborhood of the vertex k_i corresponding to i in the initial graph.

Let us note that our reduction windows fulfill equation 2. Receptive fields are still defined according to Definition 3. Let us note, however, that within the top-k framework, since reduction windows may overlap, receptive fields associated to these reduction windows also overlap and thus do not define a partition.

Proposition 7. *Let us consider A^l the adjacency matrix obtained by l iterations of equation 12 and for any vertex i surviving at level l, the set $RF^l(i)$ corresponding to the transitive closure of the hierarchical relationship defined by the reduction window. Then we have:*

$$a^l_{ij} \neq 0 \Leftrightarrow \exists u \in RF^l(i), \exists v \in RF^l(j) \quad | \quad a_{uv} \neq 0$$

Or in other terms, the receptive fields of two adjacent vertices at level l intersect (case $u = v$) or are adjacent at the base level.

Proof. **Implication :** For $l = 1$, the proof is provided by equation 13.

Let us suppose the property true at level $l-1$ and let us consider i, j defined at level l such that $a_{ij}^{l} \neq 0$. Using equation 13:

$$\exists k \in RW^{l}(i), \exists l \in RW^{l}(j) \qquad a_{kl}^{l-1} \neq 0$$

Using our recurrence hypothesis, it exists $u \in RF^{l-1}(k)$ and $v \in RF^{l-1}(v)$ such that $a_{uv} \neq 0$. Moreover:

$$RF^{l}(i) = \cup_{w \in RW^{l}(i)} RF^{l-1}(w) \text{ and } RF^{l}(j) = \cup_{w \in RW^{l}(j)} RF^{l-1}(w)$$

Hence $u \in RF^{l-1}(k) \subset RF^{l}(i)$ and in the same way $v \in RF^{l}(j)$ which concludes the proof.

Reciprocal: Since $u \in RF^{l}(i)$, it exists a sequence $i_l = i, i_{l-1}, \ldots, i_0 = u$ of children of i from i down to u. In the same way, we have a sequence $j_l = j, j_{l-1}, \ldots, j_0 = v$. Note that, a vertex may have several father. However, the set of child of a vertex is defined without ambiguity by Definition 5.

By definition of i_1 and j_1, we have $u \in RW(i_1)$ and $v \in RW(j_1)$. So we have $s_{i_1,u} > 0, s_{j_1,v} > 0$ and $a_{u,v} \neq 0$(by hypothesis). Hence $a_{i_1,j_1}^{1} \neq 0$. If we suppose the property true up to level $l-1$, we have $a_{i_{l-1},j_{l-1}}^{l-1} \neq 0$, with $i_{l-1} \in RW(i_l)$ and $j_{l-1} \in RW(j_l)$. Hence, $s_{i_l,i_{l-1}} > 0, s_{j_l,j_{l-1}} > 0$ and $a_{i_{l-1},j_{l-1}} \neq 0$. We have thus $a_{i_l,j_l} = a_{i,j} \neq 0$.

□

Remark 2. *We know, thanks to remark 1, that our reduction windows fulfill equation 2. Moreover, thanks to Proposition 7, we know that adjacent vertices at level l have adjacent receptive fields. We have thus all the required properties to apply Proposition 3. Namely, for any vertex w surviving at level l in the hierarchy:*

$$\forall (u, v) \in RF^{l}(w)^{2} \quad d_{G_0}(u, v) \leq 2 * 3^{l} - 1$$

Corollary 2. *Given two distinct vertices i and j surviving at level l, if i and j are not adjacent then their receptive fields have an empty intersection and are not adjacent.*

Proof. Using Proposition 7, we have:

$$a_{ij} = 0 \Rightarrow \forall (u, v) \in RF^{l}(i) \times RF^{l}(j) \quad a_{u,v} = 0$$

In particular, if it exists $u \in RF^{l}(i) \cap RF^{l}(j)$, we should have $a_{u,u} = 0$ which is forbidden by hypothesis. Hence $RF^{l}(i) \cap RF^{l}(j) = \emptyset$ □

Corollary 2, implies that an over-smoothing phenomenon is unlucky unless the base graph is filled with constant values, or we get a complete graph at some steps of the decimation process.

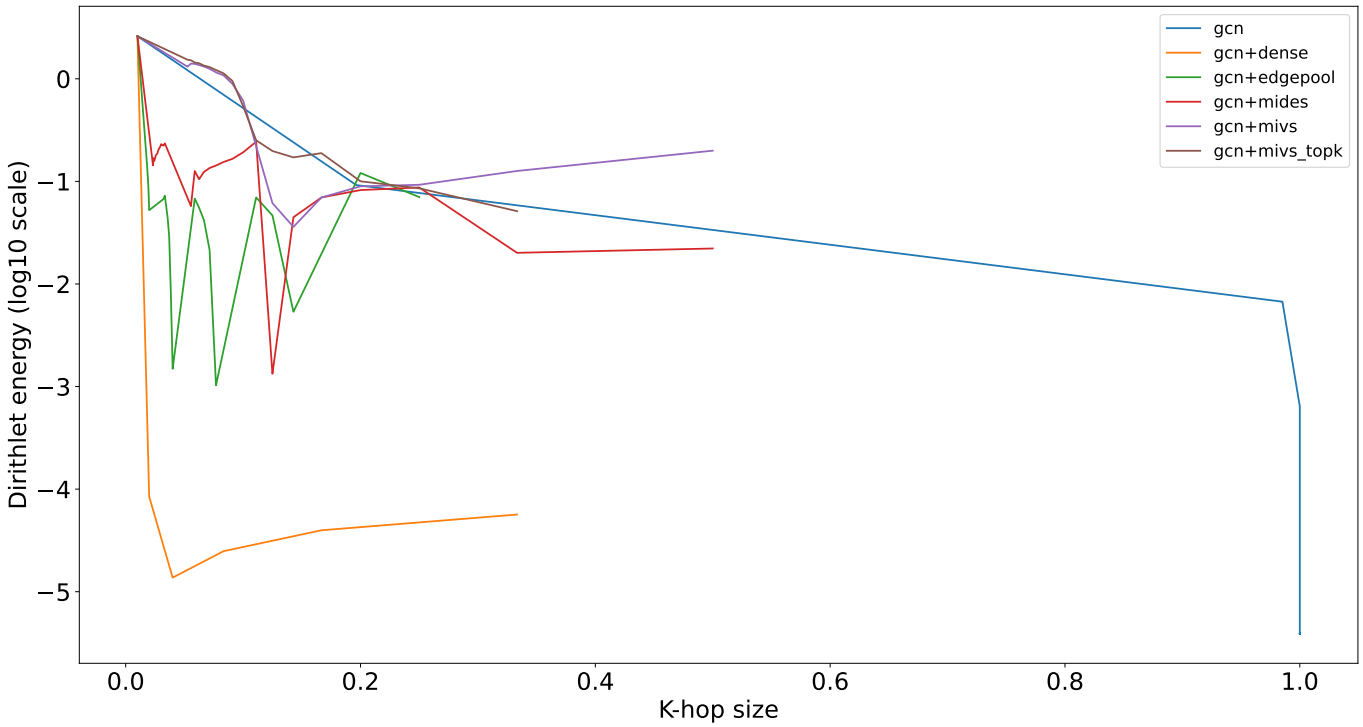

Figure 7: Let consider 1000 graphs with 100 nodes and a density of 20% generating like in [7]. The curves shows the mean Dirichlet energy according to the size of K-hop. *gcn* denotes an architecture only composed of GCN [11]. *gcn+pool* denotes an architecture alternating GCN [11] and a pooling method. *pool* can be a DiffPool [21], EdgePool [6], MIDESPool [17], MIVSPool [16] or a Top-k like in Figure 3. Note that any learnable parameters is replaced by random values.

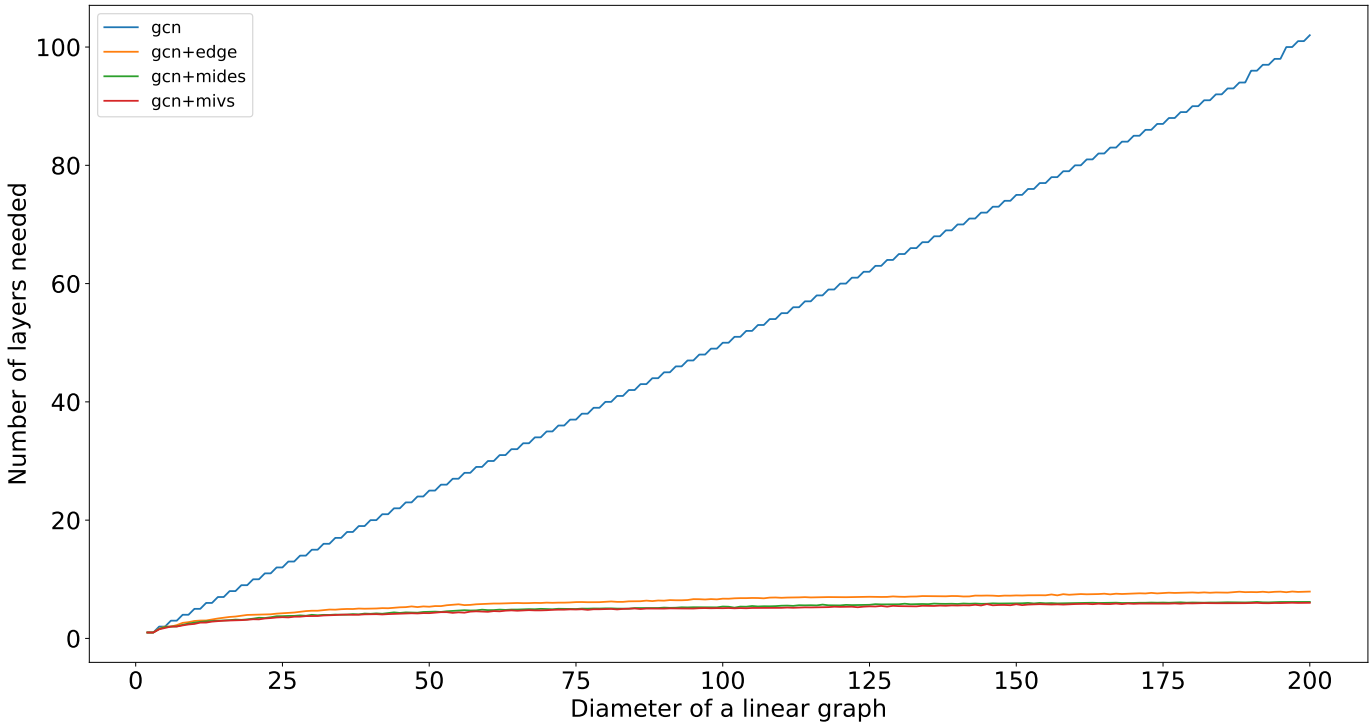

Figure 8: Let consider a linear graph with features (1,0) and (0,1) at the extreme nodes. Others nodes features are equal to (0,0). In the figure, we study the average number of layers needed to intersect the two features of the extreme nodes according to the diameter of the graph. *gcn* denotes an architecture only composed of GCN [11]. *gcn+pool* denotes an architecture alternating GCN [11] and a pooling method. *pool* can be an EdgePool [6], MIDESPool [17] or MIVSPool [16]. Note that any learnable parameters is replaced by random values.

6 Conclusion

In Section 2, we show that a sufficient low pass filtering of any signal "forces" any uniqueness set to become an allowed down-sampling. Nevertheless, for a constant signal, any sub-sampling is an allowed sub-sampling. Hence this signal point of view neglects the structure of the graph while a pooling operation must preserve both the signal content and the structural information. In Section 4, we prove that, using the pooling scheme defined in Section 3, the layer at which two distant vertices can be merged is upper bounded by a log of their distance. The access time between the features of two nodes is thus reduced in comparison with the one of graph convolution which is linear. This implies that over-squashing is less present using pooling layers. Furthermore, if each vertex from the initial graph belongs to only one receptive field in the reduced graph, then vertices in the reduced graph forms a partition of the vertices in the initial graph, applying just pooling layers like define in Definition 3 prevent to have over-smoothing. Finally, in Section 5, the case of an alternation of graph convolution and a Top-k method is studied. Under certain conditions on the selection of surviving vertices and on the construction of the reduced graph, we prove that the over-squashing and over-smoothing effects are reduced by pooling operations.

References

[1] Aamir Anis, Akshay Gadde, and Antonio Ortega. Towards a sampling theorem for signals on arbitrary graphs. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, pages 3864–3868. IEEE, 2014.

[2] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Mincut pooling in graph neural networks. *CoRR*, abs/1907.00481, 2019.

[3] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[4] George Dasoulas, Johannes F. Lutzeyer, and Michalis Vazirgiannis. Learning parametrised graph shift operators. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[5] Frederik Diehl. Edge contraction pooling for graph neural networks. *CoRR*, abs/1905.10990, 2019.

[6] Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. Towards graph pooling by edge contraction. In *ICML 2019 workshop on learning and reasoning with graph-structured data*, 2019.

[7] Paul Erdős and Alfréd Rényi. On random graphs. *Publ. math. debrecen*, 6(290-297):18, 1959.

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[9] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M. Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 7865–7885. PMLR, 2023.

[10] Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2022.

[11] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[12] Francesco Landolfi. Revisiting edge pooling in graph neural networks. In *30th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2022, Bruges, Belgium, October 5-7, 2022*, 2022.

[13] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[14] T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *CoRR*, abs/2303.10993, 2023.

[15] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[16] Stevan Stanovic, Benoit Ga"uz'ere, and Luc Brun. Maximal Independent Vertex Set applied to Graph Pooling. In *Structural and Syntactic Pattern Recognition (SSPR)*, Montr'eal, Canada, August 2022.

[17] Stevan Stanovic, Benoit Gaüzère, and Luc Brun. Maximal independent sets for pooling in graph neural networks. In Donatello Conte Pasquale Foggia, editor, *In 8th IAPR - TC-15 Workshop on Graph-based Representations in Pattern Recognition (GBR'23)*, Lecture Notes in Computer Science. Springer, September 2023. to be published.

[18] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[19] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[20] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[21] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4805–4815, 2018.

[22] Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[23] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4438–4445. AAAI Press, 2018.